3-Heights™ PDF Analysis & Repair API

Version 4.5

User Manual

Contact: pdfsupport@pdf-tools.com

Owner: **PDF Tools AG**

Kasernenstrasse 1
8184 Bachenbülach
Switzerland

http://www.pdf-tools.com

Copyright © 2001-2015

Table of Contents

1 Introduction 4

1.1 Description 4
1.2 Functions 4
Features 5
Formats 5
Compliance 5
1.3 Interfaces 6
1.4 Operating Systems 6

2 Installation 7

2.1 Windows 7
General 7
COM Interface 7
Java Interface 8
.NET Interface 8
Native C Interface 8
Uninstall, Install a New Version 8
2.2 Unix 9

3 License Management 10

3.1 Graphical License Manager Tool 10
List all installed license keys 10
Add and delete license keys 10
Display the properties of a license 11
Select between different license keys for a single product 11
3.2 Command Line License Manager Tool 11
List all installed license keys 11
Add and delete license keys 11
Select between different license keys for a single product 11
3.3 License Key Storage 11
Windows 12
Mac OS X 12
Unix / Linux 12

4 Getting Started and User's Guide 12

4.1 Overview of the API 12
What Is the 3-Heights™ PDF Analysis & Repair API About? 12
How Does the API Work? 12
4.2 Corrupt PDF Documents 14
How Do PDF Documents Get Corrupted? 14
How to Detect Corruptions? 14
What Is the Difference between Repair and Recover? 15
4.3 Concepts 16
Analysis Only 16
Analysis & Repair 17
Analysis & Conditional Repair 18
How to Use the in-Memory Functions 19

5 Programming Interfaces 20

5.1 Visual Basic 6 20

5.2 ASP VBScript 20
5.3 .NET 21
Visual Basic 21
C# 22

6 Programmer's Reference 23

6.1 The PDFRepair Interface 23
AnalysisOptions 23
Analyze 23
AnalyzeAndRepair 23
Close 23
Diagnosis 24
ErrorCode 24
ErrorLevel 24
GetFirstError 24
GetNextError 24
GetPDF 24
Open 25
OpenMem 25
RebuildOptions 25
RecoveryOptions 26
Repair 26
ReportingLevel 27
SaveAs 27
SaveInMemory 27
6.2 The PdfError Interface 28
Count 28
ErrorCode 28
Message 28
ObjectNo 28
PageNo 28
6.3 Enumerations 28
TPDFAnalysisOption 29
TPDFDiagnosis 29
TPDFErrorCode 29
TPDFPermission 29
TPDFRebuildOption 29
TPDFRecoveryOption 30

© PDF Tools AG – Premium PDF Technology

1 Introduction

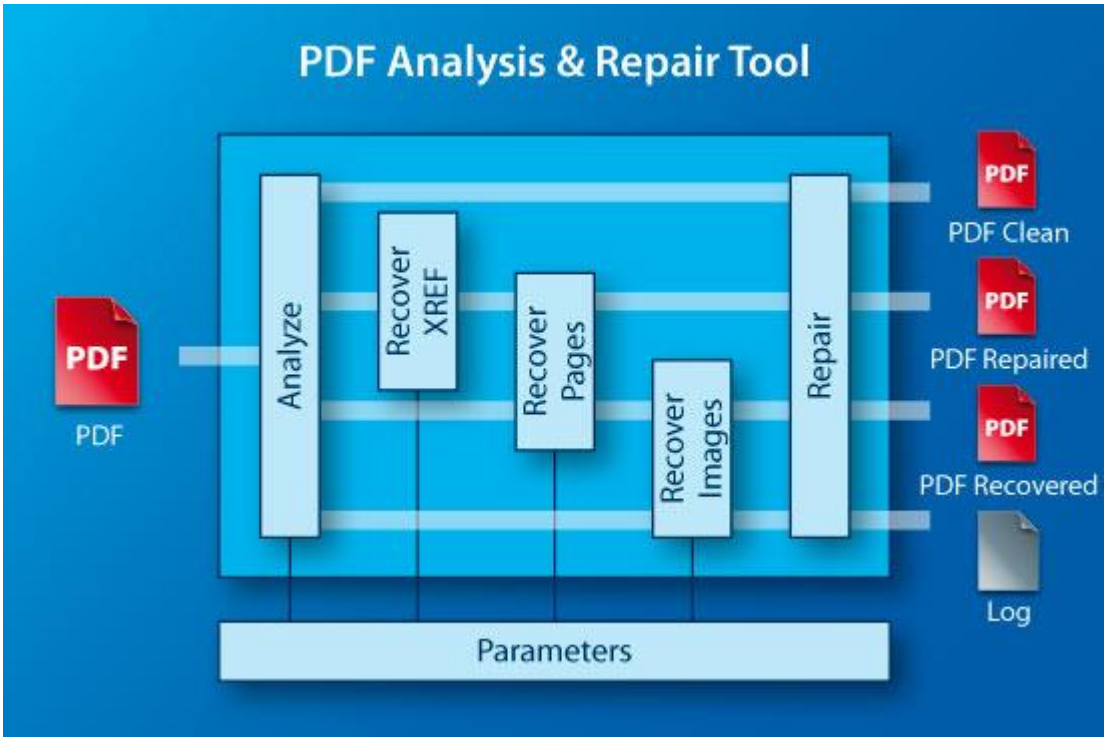
1.1 Description

The 3-Heights™ PDF Analysis & Repair tool is used to analyze, repair and restore the content of corrupt PDF documents.

Unfortunately, the number of corrupt PDF documents is incredibly huge. The cause is usually down to defective generating tools, converters and other influences such as attempts at manual editing, copying via FTP without correct settings, system crashes during PDF creation, network interruptions, defective copying on optical media, etc.

The result leads to an enormous loss of important information and to production downtimes caused by corrupt PDF documents.

The 3-Heights™ PDF Analysis & Repair analyzes PDF documents with regard to PDF specifications. Defective files are automatically repaired as far as possible and unreadable data is restored.



1.2 Functions

PDF Analysis & Repair is used to check and, where indicated, repair PDF documents. Users can determine customized profiles from a broad range of analysis and repair options. An exact and detailed description is issued for each reported error. The tool is also capable of reading and processing encrypted PDF files without any problems.

© PDF Tools AG – Premium PDF Technology

Features

Analyze and/or repair one or more PDF documents

Set analysis options, including:

- Objects
- Page tree
- Content stream

Set repair options, including:

- Restore data streams
- Restore fonts
- Restore XRef table
- Restore pages
- Restore images if pages cannot be restored

Display error description for every message, including:

- Type (errors, warnings, information)
- Error code
- Text-based description
- Page number
- Number of events

Write error messages to log file

Read encrypted PDF files

Encrypt restored file and set user authorizations

Set error level to identify whether errors, warnings or merely information occur

Set reporting level to determine which messages should be issued (errors, warnings, information)

Differentiate between Repair (corrects the errors in the document) and Restore (recreates the document based on the remaining legible information)

Formats

Input Formats

- PDF 1.x (e.g. PDF 1.4, PDF 1.5, etc.), PDF/A

Target Formats

- PDF 1.x (e.g. PDF 1.4, PDF 1.5, etc.)

Compliance

Standards: ISO 32000 (PDF 1.7)

© PDF Tools AG – Premium PDF Technology

1.3 Interfaces

The following interfaces are available: C, Java, .NET, COM.

1.4 Operating Systems

- Windows XP, Vista, 7, 8, 8.1 - 32 and 64 bit
- Windows Server 2003, 2008, 2008 R2, 2012, 2012 R2 - 32 and 64 bit
- HP-UX 11 and later PA-RISC2.0 32 bit or HP-UX 11i and later ia64 (Itanium) 64 bit
- IBM AIX 5.1 and later (64 bit)
- Linux (32 and 64 bit)
- Mac OS X 10.4 and later (32 and 64 bit)
- Sun Solaris 2.8 and later, SPARC and Intel
- FreeBSD 4.7 and later 32 bit or FreeBSD 9.3 and later 64 bit (on request)

© PDF Tools AG – Premium PDF Technology

2 Installation

2.1 Windows

The retail version of the 3-Heights™ PDF Analysis & Repair API comes as a ZIP archive containing various files including runtime binary executable code, documentation and license terms.

1. Download the ZIP archive of the product from your download account at www.pdf-tools.com.
2. Open the ZIP archive.
3. Check the appropriate option to preserve file paths (folder names) and unzip the archive to a local folder (e.g. *C:\program files\pdf-tools*).
4. The unzip process now creates the following subdirectories:
   - Bin: Contains the runtime executable binary code
   - Doc: Contains documentation files
   - Include: Contains files to include in your C / C++ project
   - Samples: Contains various samples

General

Here is an overview of the dynamic link libraries and other files that come with the 3-Heights™ PDF Analysis & Repair API:

bin\PdfRepairAPI.dll	This is the DLL that contains the main functionality (required).
*bin*NET.dll*	.NET assemblies (required when using the .NET interface).
Jar\REPA.jar	Java API archive.
Include.h*	C API include file.

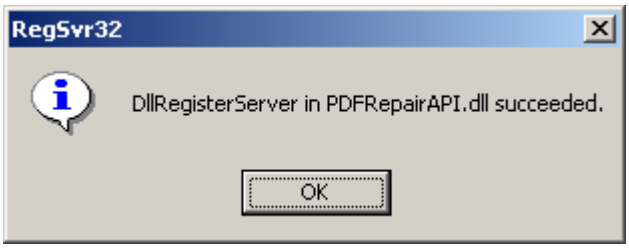
COM Interface

Before you can use the PDF Analysis & Repair Tool API component in your COM application program you have to register the component using the *regsvr32.exe* program that is provided with the Windows operating system in the directory System32. The following screenshot shows the registration of the PDF Analysis & Repair API DLL:

On Windows Vista an later, the command must be executed in an Command Prompt that runs as Administrator.

When using a Windows 64 bit platform, and you want to register the 32 bit version of the 3-Heights™ PDF Analysis & Repair API, it is required to use the regsvr32 from the WOW64 directory instead of System32.

If the registration process succeeds the following box is displayed:



The installation process is now complete.

Java Interface

When using the Java interface, the Java-wrapper *jar\REPA.jar* needs to be on the CLASSPATH. *bin\PdfRepairAPI.dll* needs to be on the PATH.

.NET Interface

The path where the *PDFRepairAPI.dll* resides must be known to the project, e.g. should be added to the environment variable "PATH".

The .NET assemblies (**NET.dll*) are to be added as references. See also chapter *"Programming Interfaces" -> ".NET"*.

Native C Interface

The header file *pdfrepairapi_c.h* needs to be included in the C program. The library *lib\PdfRepairAPI.lib* needs to be linked to the project. The dynamic link library *bin/PdfRepairAPI.dll* needs to be in path of executables (e.g. on the environment variable "PATH").

Uninstall, Install a New Version

In order to uninstall the product, undo all the steps done during installation, e.g. un-register using regsvr32 –u, delete all files, etc.

Note that an expired evaluation DLL cannot be unregistered. If you would like to un-register an expired evaluation DLL, download a new (non-expired) evaluation version, overwrite the old version and un-register it.

Installing a new version does not require to uninstall the previously installed old version. The files of the old version can directly be overwritten with the new version. If you use the COM interface, you must register the new DLL, un-registering the old version is not required.

© PDF Tools AG – Premium PDF Technology

2.2 Unix

Here is an overview of the shared libraries and other files that come with the 3-Heights™ PDF Repair API:

bin/PdfRepairAPI.so	This is the shared library that contains the main functionality.
jar/REPA.jar	Java API archive.
include/.h*	C API include file.

- Unpack the archive in an installation directory, e.g. */usr/pdftools.com/*

  Copy or link the shared object into one of the standard library directories, e.g:

  ```
  ln -s /usr/pdftools.com/bin/libPdfRepairAPI.so /usr/lib
  ```

- In case you have not yet installed the GNU shared libraries, get a copy of these from http://www.pdf-tools.com; extract the shared images and copy or link them into */usr/lib* or */usr/local/lib*.

  On Mac OS/X platforms, the shared library must have the extension .jnilib for use with Java. We suggest that you create a file link for this purpose by using the following command:

  ```
  ln libPdfRepairAPI.dylib libPdfRepairAPI.jnilib
  ```

© PDF Tools AG – Premium PDF Technology

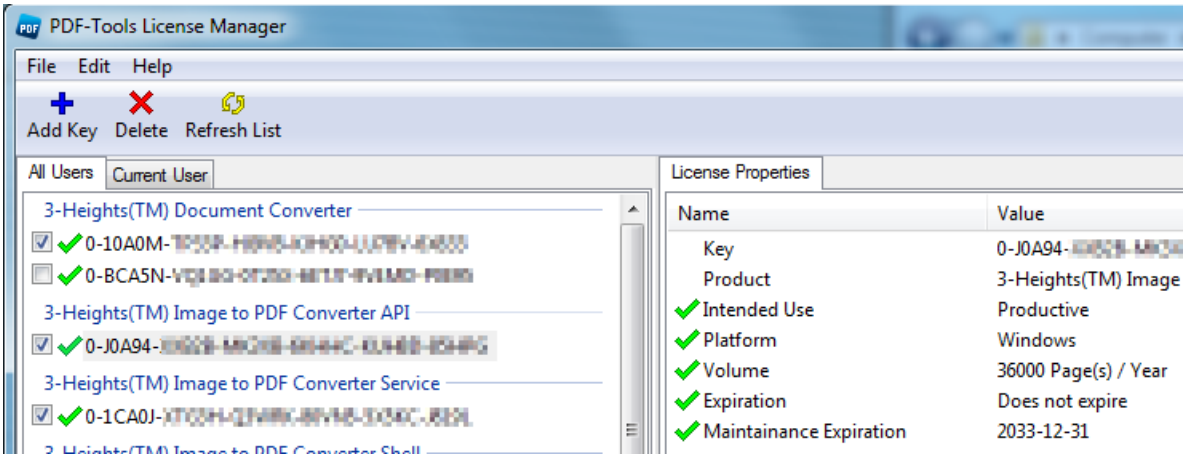
3 License Management

There are three possibilities to pass the license key to the application:

1. The license key is installed using the GUI tool (Graphical user interface). This is the easiest way if the licenses are managed manually. It is only available on Windows.
2. The license key is installed using the shell tool. This is the preferred solution for all non-Windows systems and for automated license management.
3. The license key is passed to the application at runtime via the "LicenseKey" property. This is the preferred solution for OEM scenarios.

3.1 Graphical License Manager Tool

The GUI tool *LicenseManager.exe* is located in the *bin* directory of the product kit.



List all installed license keys

The license manager always shows a list of all installed license keys in the left pane of the window. This includes licenses of other PDF Tools products.

The user can choose between:

- Licenses available for all users. Administrator rights are needed for modifications.
- Licenses available for the current user only.

Add and delete license keys

License keys can be added or deleted with the "Add Key" and "Delete" buttons in the toolbar.

- The "Add key" button installs the license key into the currently selected list.
- The "Delete" button deletes the currently selected license keys.

Display the properties of a license

If a license is selected in the license list, its properties are displayed in the right pane of the window.

Select between different license keys for a single product

More than one license key can be installed for a specific product. The checkbox on the left side in the license list marks the currently active license key.

3.2 Command Line License Manager Tool

The command line license manager tool *licmgr* is available in the *bin* directory for all platforms except Windows.

A complete description of all commands and options can be obtained by running the program without parameters:

```
licmgr
```

List all installed license keys

```
licmgr list
```

The currently active license for a specific product ist marked with a star '*' on the left side.

Add and delete license keys

Install new license key

```
licmgr store X-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX
```

Delete old license key

```
licmgr delete X-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX
```

Both commands have the optional argument `-s` that defines the scope of the action:

- `g`: For all users
- `u`: Current user

Select between different license keys for a single product

```
licmgr select X-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX
```

3.3 License Key Storage

Depending on the platform the license management system uses different stores for the license keys.

Windows

The license keys are stored in the registry:

- HKLM\Software\PDF Tools AG (for all users)
- HKCU\Software\PDF Tools AG (for the current user)

Mac OS X

The license keys are stored in the file system:

- /Library/Application Support/PDF Tools AG (for all users)
- ~/Library/Application Support/PDF Tools AG (for the current user)

Unix / Linux

The license keys are stored in the file system:

- /etc/opt/pdf-tools (for all users)
- ~/.pdf-tools (for the current user)

Note: The user, group and permissions of those directories are set explicitly by the license manager tool.

It may be necessary to change permissions to make the licenses readable for all users. Example:

```
chmod -R go+rx /etc/opt/pdf-tools
```

4 Getting Started and User's Guide

4.1 Overview of the API

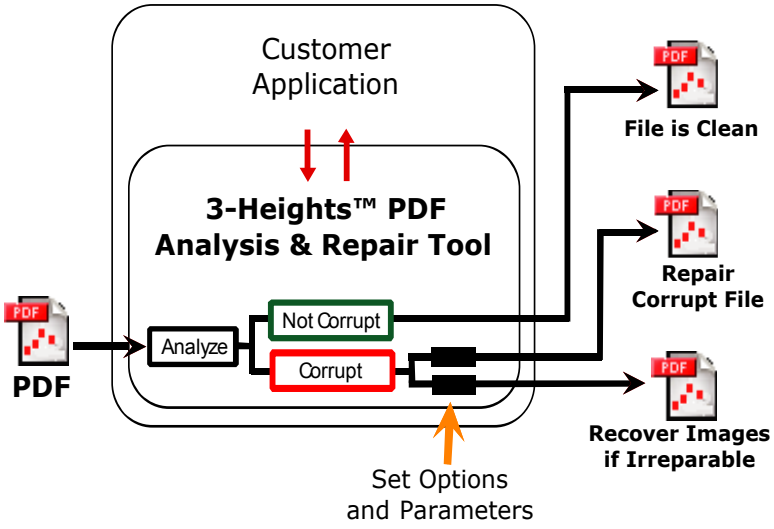
What Is the 3-Heights™ PDF Analysis & Repair API About?

The API provides two main functionalities as its name indicates:

1. Analysis of PDF documents, detect and report corruptions
2. Repair or recover the detected corruptions and save the result in a new PDF document

How Does the API Work?

The API requires a PDF document as input. In this manual this document is referred to as input-document. In the graphic below, that's the PDF on the left hand side.



- In a first step the input-document is **opened** for reading. This is done using the function *Open*.
- In the next step **analysis options** are set using the property *AnalysisOptions*. The more analysis options are set, the deeper the analysis goes and the longer it takes.
- In case the document is going to be repaired later on, the **recovery options** can be set at this point using the property *RecoveryOptions*. Recovery options can also be set after the analysis and before the repair step. However as we will see, these two steps can be combined, therefore it make sense to set them previously.
- The document is being **analyzed** using either the function *Analysis* or *AnalysisAndRepair*. As a result of the analysis, the document is qualified valid or corrupt.
- For corrupt documents, all corrption can be listed in an **error report**. In the case of an analysis-only process, the input-document can now be closed and the process is done.
- If the process is also to repair the document, it is **saved as a new PDF** document using the *SaveAs* function. If the user chooses to repair the PDF document, a new PDF document is created. This document is referred to as output-document. The output-document is completely rebuilt from scratch using all readable information from the input-document. This means if a valid PDF document is repaired, a new document is created as well.

  If a PDF document is corrupt, the PDF can either be repaired or recovered depending on the level of corruption. The difference between repairing and recovering is described in a separate chapter.
- The input-document is **closed** using *Close*.

See also chapter *"Concepts".*

4.2 Corrupt PDF Documents

How Do PDF Documents Get Corrupted?

One needs to be aware that PDF is a complex format, its specification is more than 1000 pages. Within PDF there are embedded objects, such as different types of fonts, images or compressions, which again are complex on their own and have specifications that are even larger than the PDF specification itself.

There are uncountable different PDF products available, and virtually none of them is capable to support everything PDF offers. And only few of them create actually valid PDF. Most freeware, or home-made PDF creators have flaws. These flaws are often not detected initially simply because the widely used PDF viewer applications detect and repair these errors on the fly. The creator of the PDF doesn't even notice his PDF is corrupt, because the PDF viewer application fixes or ignores the problem silently. A creator often does not have the goal to create a PDF, but just a PDF which can be viewed.

Reason 1: Incorrect PDF creators

PDF is a binary format. Most of its content is compressed. Editing a PDF file with a text editor, or transmitting a PDF in text mode instead of binary mode (e.g. FTP) corrupts the PDF. Partially transmitting a PDF file cuts off part of the document, this loss of information is not recoverable.

Reason 2: Binary file is damaged

There are further reasons, but the two reasons mentioned are certainly the most common.

How to Detect Corruptions?

The most obvious way to detect a problem with a PDF document is if it doesn't open a PDF viewer application, or there is an error message when opening the document, or part of the document cannot be displayed correctly. For most user this is the only situation where they actually are aware the document is corrupt. Any other corruption that has no direct impact to viewing the document is often ignored.

If documents are being archived or must be of good quality for other reasons, they can be analyzed using a PDF analysis tool.

The 3-Heights™ PDF Analysis & Repair API analyzes documents and detects whether they are valid or not according to the PDF specification.

A simpler test to see whether a document is valid or not is to open it in Adobe Acrobat Professional and close it again. If one is prompted to save the document, it can be an indication that the document was corrupt and was repaired and the repaired document is now displayed to the user. This test however does not provide any information about what was corrupt, i.e. what was repaired. The save-prompt could also be unrelated to corruptions, but be of another nature, such as a Java script.

© PDF Tools AG – Premium PDF Technology

What Is the Difference between Repair and Recover?

The 3-Heights™ PDF Analysis & Repair API can repair virtually everything it can detect. It can however not recover lost information. E.g. if a PDF document was sent via E-Mail attachment and only have of the attachment was sent before the connection was cut, information is lost. If information is lost, a document cannot be transformed back to its original state. In this case the document can only be **recovered**, meaning all the remaining information contained in the PDF document is recovered and used to create a new, valid PDF, however the new PDF is different from the original.

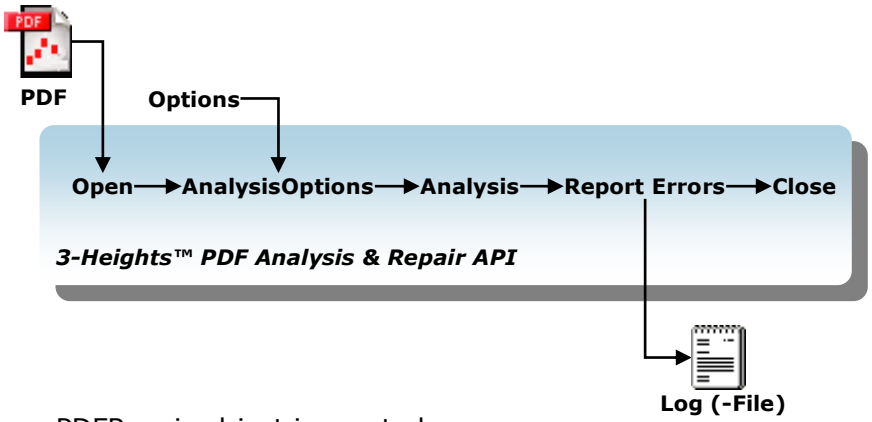
If a document contains syntactic or semantic faults which can be detected and fixed, the document can be **repaired**. An trivial example for such a case is image which contains image data with a length of 100 bits, the Length attribute of the image object however states a different, incorrect value, then this value can be corrected and the document can be repaired.

© PDF Tools AG – Premium PDF Technology

4.3 Concepts

Analysis Only

The process for analysis-only is the most simply process that can be implemented by the API. The steps in this process are shown in the graphic below:



- A new PDFRepair object is created.
- A PDF input-document is **opened** using the function *Open*.
- **Analysis-options** are set using the property *AnalysisOptions*. This step is optional.
- The **analysis** of the input-document is performed using the function *Analysis*.
- A **list of error** objects can be retrieved using the functions *GetFirstError* and consecutive calls to the function *GetNextError* until no more errors are returned. An error objects provides information about a corruption error, such as an error code or an error message.
- The input-document is then **closed** again using the function *Close*.

The call sequence for analysis-only is:

- `Create Object`
- `Open`
- `Set AnalysisOptions (Optional)`
- `Analysis`
- `Report Errors (Optional)`
- `Close`

A simplified Visual Basic 6 program with the above call sequence looks about as shown below:

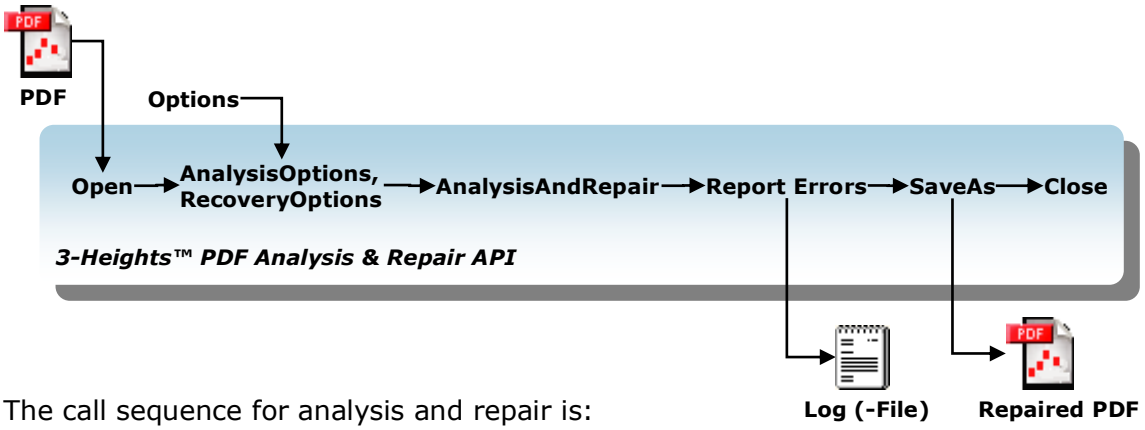
- `Dim repair As New PDFREPAIRAPILib.PDFRepair`

- `repair.Open(...)`
- `repair.AnalysisOptions = ...`
- `repair.Analysis()`
- ```
  ' Report Errors
  Dim Err As PDFREPAIRAPILib.PdfError
  Set Err = repair.GetFirstError
  While (Not Err Is Nothing)
      ' Do something with the error, e.g. output Err.Message
      ...
      Set Err = repair.GetNextError
  Wend
  ```
- `repair.Close()`

A more detailed and executable Visual Basic 6 sample is provided with the release as well as with the evaluation version.

Analysis & Repair

Often corrupt documents not only need to be detected, but also repaired or recovered. As opposed to the analysis-only process, here the file is analyzed and repaired in one step using the function *AnalysisAndRepair*. The repaired document is saved as a new document using the function *SaveAs*.

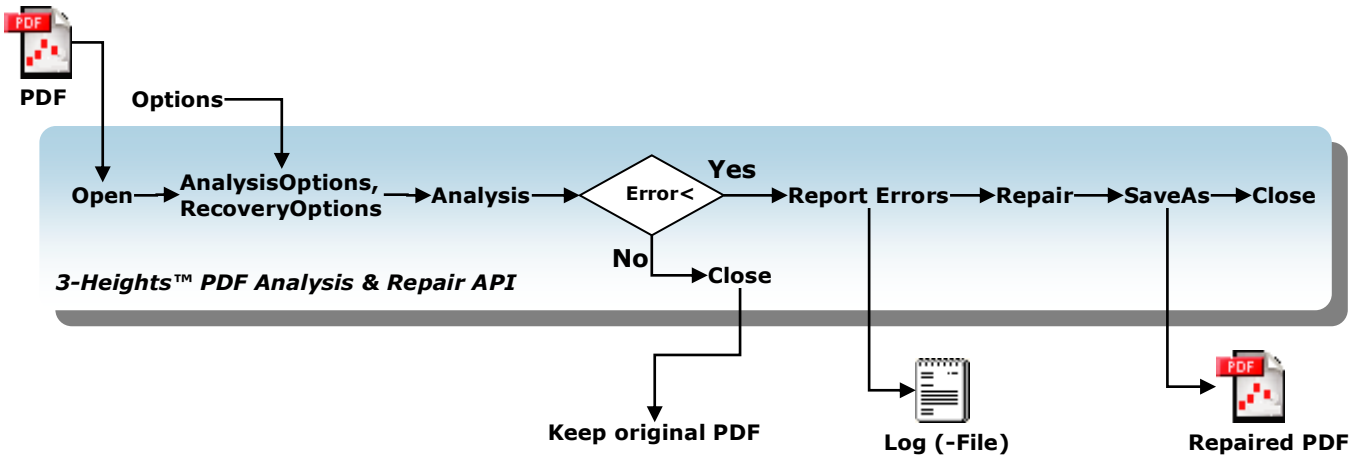


The call sequence for analysis and repair is:

- `Open`
- `Set AnalysisOptions and RecoveryOptions (Optional)`
- `AnalysisAndRepair`
- `Report Errors (Optional)`
- `SaveAs`
- `Close`
  ```

Analysis & Conditional Repair

In the Analysis & Repair process using *AnalysisAndRepair*, every document is repaired, not matter the analysis. A more sophisticated approach is to separate these two step to first analyze the document and only repair it, if corruptions are actually detected.



The call sequence for analysis and conditional repair is:

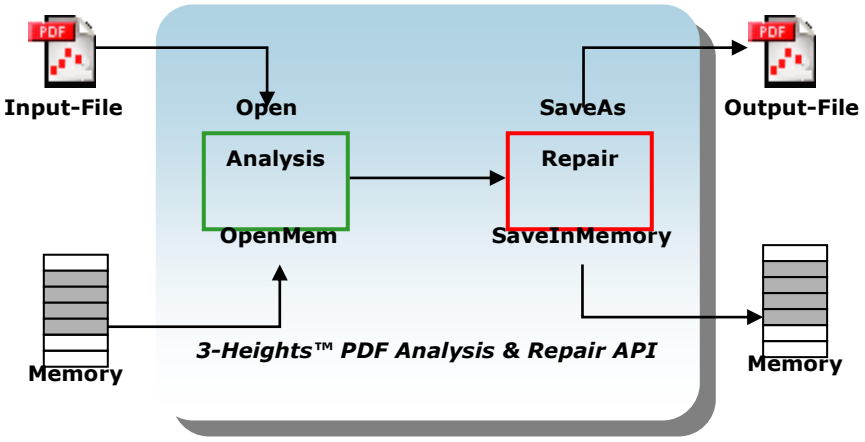
- `Create Object`
- `Open`
- `Set AnalysisOptions and RecoveryOptions (Optional)`
- `Analysis`
- `If Errors`
  - `Report Errors (Optional)`
  - `Repair`
  - `SaveAs`
- `Close`

How to Use the in-Memory Functions

The 3-Heights™ PDF Analysis & Repair API always requires a PDF input-document from which it reads and optionally a PDF output-document to where the result is saved to.

To open and save to from files, the functions *Open* and *SaveAs* are used.

Instead of accessing files, the PDF documents can be read and written to in-memory. The corresponding functions are *OpenMem* and *SaveInMemory*.



Once the output-document is saved to memory using *SaveInMemory*, that memory block can be accessed using the function *GetPDF*.

A call sequence to create a first PDFRepair object that opens a PDF from file and stores its output in-memory and then a second object, which reads that in-memory document and saves it back to a file looks like this:

```
PDFRepair1.Open(InputFile)

PDFRepair1.SaveInMemory()

PDFRepair1.Close()

PDFRepair2.OpenMem(PDFRepair1.GetPDF())

PDFRepair2.SaveAs(OutputFile)

PDFRepair2.Close()
```

This call sequence of course does not make much sense, it's merely used to illustrate how to use of the in-memory functions. In a real application, the in-memory document is read from another application or a database.

5 Programming Interfaces

5.1 Visual Basic 6

After installing the 3-Heights™ PDF Analysis & Repair API and registering the COM interface (see chapter *"COM Interface"*), you find a Visual Basic example *repair.vbp* in the directory *samples/VB/*. You can either use this sample as a base for an application, or you can start from scratch.

The 3-Heights PDF Analysis & Repair Tool API is very easy to use.

A Visual Basic 6 sample looks as simple as this:

```
Private Sub repair_Click()
      Dim repair As New PDFREPAIRAPILib.PDFRepair
      repair.repair "C:\input.pdf", "C:\output.pdf", "C:\log.txt"
End Sub
```

If a PDF document cannot be repaired and contains images, it is possible to recover the images by setting the property:

```
repair.RecoveryOptions = repair.RecoveryOptions or eRecoverImages
```

5.2 ASP VBScript

The class name to be used is "PDFREPAIRAPI.PDFRepair".

Simplified example:

```
<%@ Language=VBScript %>
<%
    option explicit
    dim repair
    set repair = Server.CreateObject("PDFREPAIRAPI.PDFRepair")
    repair.Open("path\file_to_be_repaired.pdf")
    repair.AnalyzeAndRepair()
    repair.SaveAs("path\output_file.pdf")
    repair.Close
%>
```

© PDF Tools AG – Premium PDF Technology

5.3 .NET

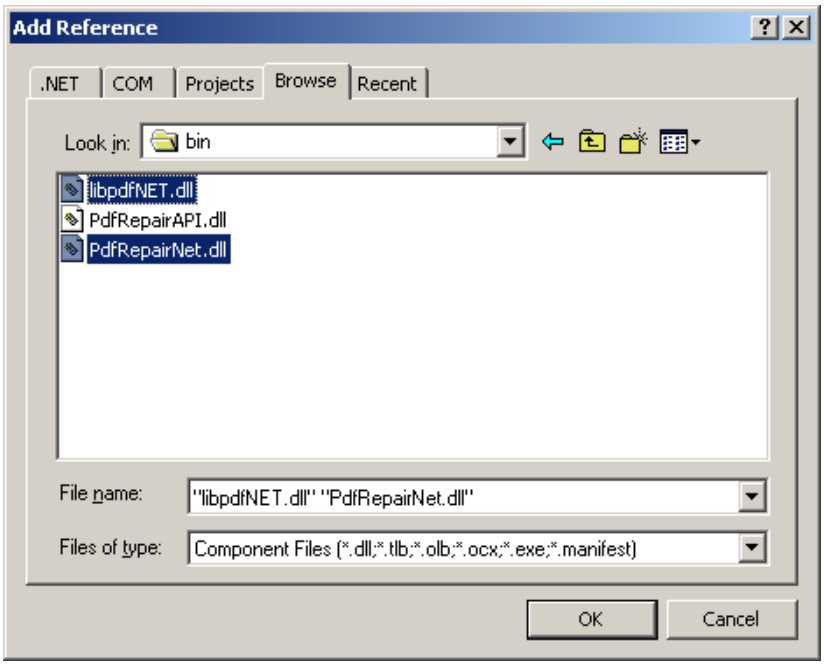
As opposed to previous versions, the Windows build numbers 1.7.1.* and later provide a .NET interface.

There should be at least one .NET sample for MS Visual Studio 2005 available in the ZIP archive of the Windows Version of the 3-Heights™ PDF Security API. Easiest for a quick start is to refer to this sample.

In order to create a new project from scratch, do the following steps:

1. Start Visual Studio and create a new C# or VB project.
2. Add a reference to the .NET assemblies.

   To do so, in the "Solution Explorer" right-click your project and select "Add Reference…". The "Add Reference" dialog will appear. In the tab "Browse", browse for the .NET assemblies *libpdfNET.dll* and *PdfRepairNET.dll* add them to the project as shown below:



3. Import namespaces (Note: This step is optional, but useful.)
4. Write Code

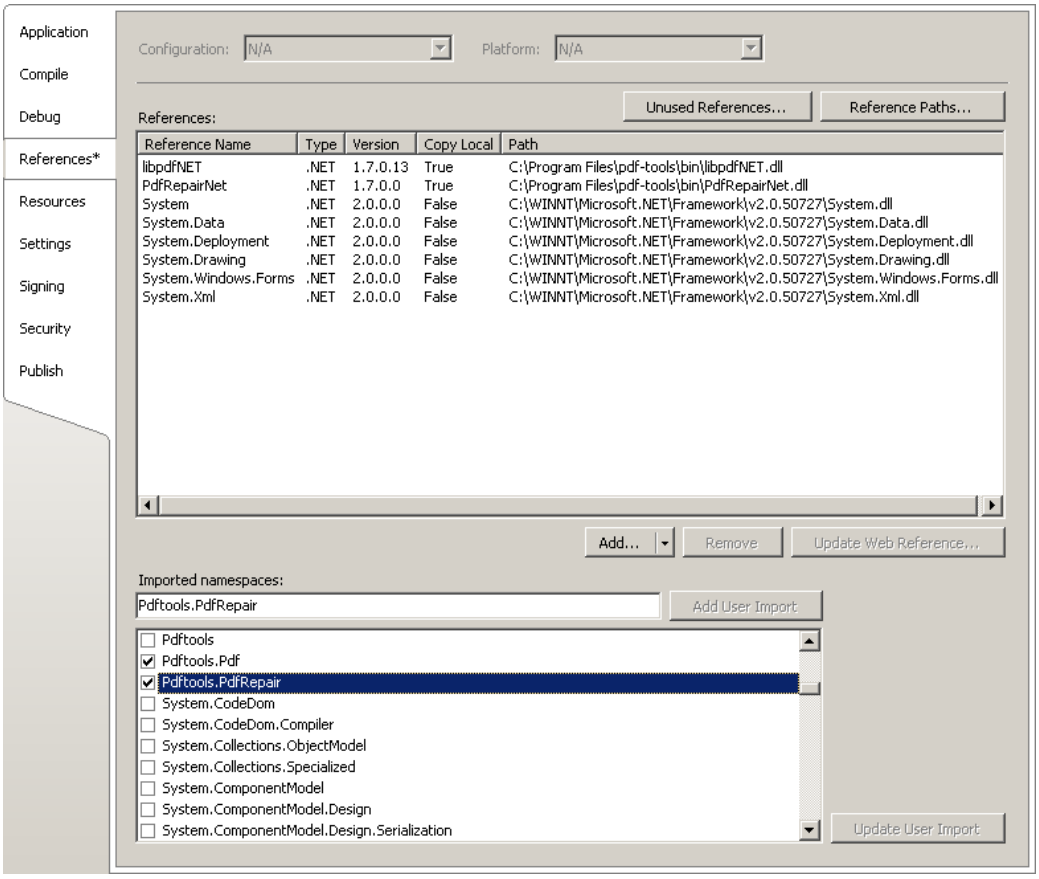
   Steps 3 and 4 are shown separately for C# and Visual Basic.

Visual Basic

3. Double-click "My Project" to view its properties. On the left hand side, select the menu "References". The .NET assemblies you added before should show up in the upper window.

   In the lower window import the two namespaces *Pdftools.Pdf* and *Pdftools.PdfRepair*.

You should now have settings similar as in the screenshot below:



4. The class Pdftools.PdfRepair.Repair can now be used as shown in the code snippet below:

```
Dim doc As New Pdftools.PdfRepair.Repair

' Or if the namespace Pdftools.PdfRepair is imported:

' Dim doc As New Repair

doc.Open(...)

...
```

C#

3. Add the following namespaces:

```
using Pdftools.Pdf;

using Pdftools.PdfRepair;
```

4. The class Pdftools.PdfRepair.Repair can now be used as shown in the code snippet below:

```
Repair doc = new Repair();

doc.Open(...)

...
```

6 Programmer's Reference

Note this manual describes the COM interface only. Other interfaces (C, Java, .NET) however work similarly, i.e. they have calls with similar names and the call sequence to be used is the same as with COM.

6.1 The PDFRepair Interface

AnalysisOptions

```
Property TPDFAnalysisOption AnalysisOptions
Accessors: Get, Set
Default: eAnalyzeObjects + eAnalyzePageTree + eAnalyzeContentStreams
```

This property sets the analysis options. Options can be turned off to increase the speed of the analysis. See enumeration *TPDFAnalysisOption*.

Analyze

```
Method Boolean Analyze()
```

This method analyzes the input document for errors.

- Return value:

  True: The Analysis was successful.

  False otherwise.

AnalyzeAndRepair

```
Method Boolean AnalyzeAndRepair()
```

This method analyzes the input document and creates a repaired output document.

- Return value:

  True: The Analysis was successful. And a repaired document could be created.

  False otherwise.

Close

```
Method Boolean Close()
```

Close the input file.

- Return value:

  True: The input file could be closed.

  False otherwise.

Diagnosis

```
Property TPDFDiagnosis Diagnosis
Accessors: Get
```

This property returns the diagnosis flags. See also enumeration *TPDFDiagnosis*.

ErrorCode

```
Property TPDFErrorCode ErrorCode
Accessors: Get
```

This property can be accessed to receive the latest error code. See enumeration *TPDFErrorCode*.

ErrorLevel

```
Property Integer ErrorLevel
Accessors: Get
```

This property can be accessed to check whether no errors (0), warnings only (1) or errors (2) were found during the analysis. This property should be get after *Analyze()*.

GetFirstError

```
Method TPDFErrorCode GetFirstError()
```

This method returns the first error, it can also be a warning.

- Return value:

  The first error if there are any.

  Nothing otherwise.

GetNextError

```
Method TPDFErrorCode GetNextError()
```

This method returns the next error, it can also be a warning.

- Return value:

  The next error if there are any.

  Nothing otherwise.

GetPDF

```
Method Variant GetPDF()
```

This method returns the PDF which was previously saved to memory using the method *SaveInMemory()*. This method must be used after *Close()*.

- Return value:

A 1-dimensional byte array containing the PDF document.

Open

```
Method Boolean Open(String FileName, String Password)
```

Opens the input file.

- Parameters

  String InputFile: the file name and optionally the file path, drive or server string according to the operating systems file name specification rules of the input file.

  String Password (optional): the user or the owner password of the encrypted PDF document. If this parameter is left out an empty string is used as a default.

- Return value:

  True: The file could successfully be opened.

  False: The file does not exist, is corrupt, or the password is not valid.

OpenMem

```
Method Boolean OpenMem(Variant MemBlock, String Password)
```

This method opens a PDF memory block, i.e. makes the objects contained in the PDF document accessible. If the document is already open it is closed first.

- Parameters:

  MemBlock: The memory block containing the PDF file given as a one dimensional byte array.

  Password (optional): The user or the owner password of the encrypted PDF document. If this parameter is left out an empty string is used by default.

- Return value:

  True: The document could successfully be opened.

  False: The document does not exist, is corrupt, or the password is invalid.

RebuildOptions

```
Property TPDFRebuildOption RebuildOptions
Accessors: Get, Set
Default: 0
```

This property can be used to get or set the rebuild options.

- eRebuildStreams

  Recompress all streams. This setting is recommended for files with corrupt streams.

- eRebuildFonts

Rebuild fonts.

- eRebuildFontsAsType1

  Convert Compact Font Format (CFF) simple fonts to Type1. This property can be used together with eRebuildFonts only.

RecoveryOptions

```
Property TPDFRecoveryOption RecoveryOptions
Accessors: Get, Set
Default: eRecoverXREF + eRecoverPages
```

This property can be used to get or set the recovery options.

- eRecoverXREF

  Don't recover the XREF table. This option is useful if processing a document takes too long, since repairing the cross-reference table is very time consuming.

- eRecoverPages

  If pages are not part of the page tree (loose pages), they will be recovered and added at the end of the document. If they should not be recovered, these pages will be removed from the document.

- eRecoverImages

  This property defines if the 3-Heights™ PDF Analysis & Repair Tool should try to recover images of a PDF document if the document cannot be repaired.

Repair

```
Method Boolean Repair(String InputFile, String OutputFile, String
LogFile)
```

This method opens a PDF file, analyzes and repairs it. The repaired file is saved with a new file name. Optionally a log file can be generated.

- Parameters

  String InputFile: the file name and optionally the file path, drive or server string according to the operating systems file name specification rules of the input file.

  String OutputFile: the file name and optionally the file path, drive or server string of the output file (the repaired file).

  String LogFile (optional): The path to the log-file.

- Return value:

  True: The file could successfully be processed

  False: The file could not be processed and therefore not be repaired

ReportingLevel

```
Property Integer ReportingLevel
Accessors: Get, Set
Default: 3
```

With this property the reporting level can be set or get. The supported levels are:

0	none	Nothing is reported
1	errors	Errors are reported
2	warnings	Errors and warnings are reported
3	information	Error, warnings and information are reported

The property ReportingLevel must be set before the Open Method in order to be applied.

SaveAs

```
Method Boolean SaveAs(String FileName, String UserPw, String OwnerPw,
Long PermissionFlags)
```

This method saves the document to a file.

- Parameters

  String InputFile: The file name and optionally the file path, drive or server string according to the operating systems file name specification rules of the input file.

  String UserPw (optional): Set the user password of the PDF document. If this parameter is omitted, the default password is used. Use 0 to set no password.

  String OwnerPw (optional): Set the owner password of the PDF document. If this parameter is omitted, the default password is used. Use 0 to set no password.

  Long PermissionFlags (optional): The permission flags. The permissions that can be granted are listed in the enumeration TPDFPermission.

  To not encrypt the output document, set PermissionFlags to –1, user and owner password to 0.

  In order to allow high quality printing, flags ePermPrint and ePermDigitalPrint need to be set.

- Return value:

  True: The file could successfully be created.

  False otherwise.

SaveInMemory

```
Method Boolean SaveInMemory()
```

This method saves the output PDF in memory. After the *Close()* call it can be accessed using the method *GetPDF()*.

6.2 The PdfError Interface

Count

```
Property Long Count
Accessors: Get
```

This property returns how many times the error occurs on the page.

ErrorCode

```
Property TPDFErrorCode ErrorCode
Accessors: Get
```

This property returns the error code. See enumeration *TPDFErrorCode*.

Message

```
Property String Message
Accessors: Get
```

This property returns an explaining error message.

ObjectNo

```
Property Long ObjectNo
Accessors: Get
```

This property is not yet supported.

This property returns the object number at which the error occurs. If the error is not related to a particular object, 0 is returned.

PageNo

```
Property Long PageNo
Accessors: Get
```

This property returns the page number on which the error occurs. If the error is not related to a particular page number, 0 is returned.

6.3 Enumerations

Note: Depending on the interface, enumerations may have "TPDF" as prefix (COM, C) or "PDF" as prefix (.NET) or no prefix at all (Java).

© PDF Tools AG – Premium PDF Technology

TPDFAnalysisOption

eAnalyzeObjects	Analyze objects
eAnalyzePageTree	Analyze page tree
eAnalyzeContentStreams	Analyze content streams

TPDFDiagnosis

eDiagnosisOpen	Diagnose opening
eDiagnosisObjects	Diagnose objects
eDiagnosisPages	Diagnose pages

TPDFErrorCode

All TPDFErrorCode enumerations start with "PDF_" followed by a single letter which is one of "S", "E", "W" or "I", an underscore and a descriptive text. The single letter gives in an indication of the type of error. These are: **S**uccess, **E**rror, **W**arning, **I**nformation. With respect to corrupt PDF files: An error indicates a corruption in the PDF, the file may or may not be readable. A warning indicates the file is readable but not valid.

A full list of all PDF Tools error codes is available in the header file *pdferror.h*. The error codes that are listed to file access are listed here.

PDF_S_SUCCESS	The operation was completed successfully.
LIC_E_NOTINIT, … LIC_E_LEVEL	Various license management related errors.
PDF_E_FILEOPEN	The file couldn't be opened.
PDF_E_FILECREATE	The file couldn't be created.
PDF_E_PASSWORD	The authentication failed due to a wrong password.

TPDFPermission

ePermPrint	Low resolution printing
ePermModify	Changing the document
ePermCopy	Content copying or extraction
ePermAnnotate	Annotations
ePermFillForms	Filling of form fields
ePermSupportDisabilities	Support for disabilities
ePermAssemble	Document Assembly
ePermDigitalPrint	High resolution printing

TPDFRebuildOption

eRebuildStreams	Rebuild (recompress) streams

© PDF Tools AG – Premium PDF Technology

eRebuildFonts	Rebuild fonts
eRebuildFontsAsType1	Rebuild and convert CFF simple fonts to Type1

TPDFRecoveryOption

eRecoverXREF	Do not recover X-REF table
eRecoverPages	Recover pages
eRecoverImages	Recover images

© PDF Tools AG – Premium PDF Technology